Zebra DevCon 2023

ZEBRA

Printer Command Languages

How to Speak "Printer"

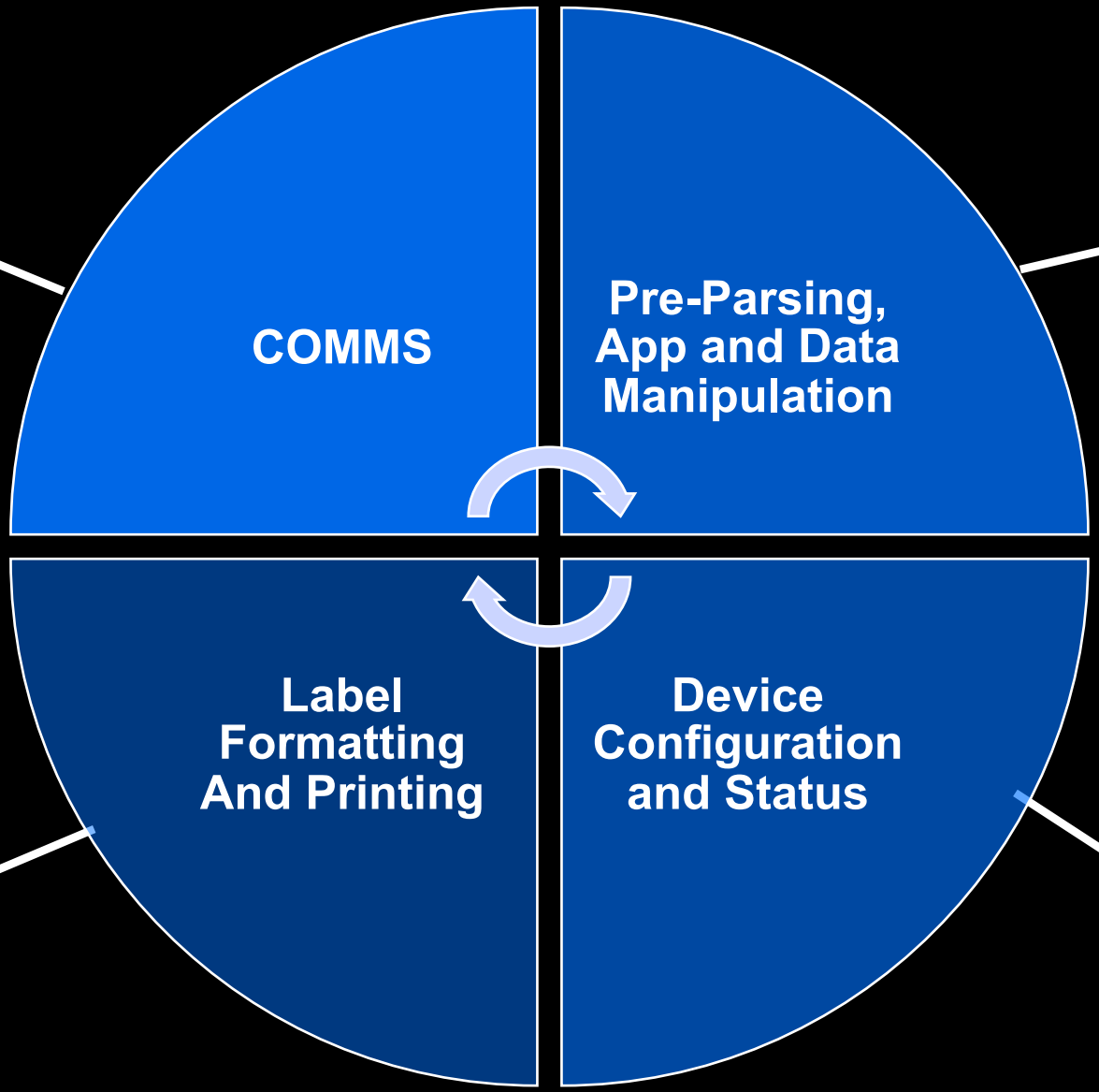
Leo Lowy
Director, Sensor Product Management

Print DNA Printers

Zebra DevCon 2023

Printer OS

Zebra DevCon 2023



RS232, Parallel, USB Client/Host/OTG, Ethernet, Wi-Fi, Bluetooth, WAN (future)

TCP/UDP, HTTPS, POP3, SMTP, Web Sockets

ZBI
Java (future)

Licensing System

ZPL, EPL, CPCL, XML, Emulated Printer Languages

Fonts, Unicode BMP & PNG, DPI Media Presentation

ZPL, SDG, OIDs, MQTT, JSON

Command Parsing, Multi-Lang support

Printer Command Languages: What To Use

Zebra DevCon 2023

- **Zebra Programing Language (ZPL)**
  - 321 Commands
- **Set-Get-Do (SGD)**
  - 1100 Commands
- **Eltron Programming Language**
  - 86 Commands
- **Comtec Programming Language**
  - 148 commands
- **Java Scrip Notation (JSON) with SGD**
  - Multiple Purposes for Multiple Languages
- **XML**
  - Extensible Markup Language
- **Simple Network Monitoring Protocol**
  - 1019 Object Identifiers (OIDs)
- **Zebra Basic Interpreter**
  - 92 Commands
- **Emulations**
  - Multiple Command Languages

Formatting & Encoding

Config & Status

When Required

As Needed

Zebra Programming Language (ZPL)

Zebra DevCon 2023

- In the Printer OS since 1988
- Use for multiple tasks:
  - Used to print fonts, graphics, barcodes, encode RFID tags
  - Printer configuration and printer status checking
- Not case sensitive.
- Commands are 2 characters and are preceded by a command prefix (^/~ by default), with params comma separated
  - Many parameters have defaults
- Backwards compatibility controls the creation of new commands and parameters
  - Commands exist that are similar but have expanded functionality to maintain compatibility

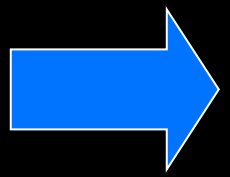
Using ZPL – It's All About the Dots

Zebra DevCon 2023

- All X & Y locations are measured in printer dots.
- Text begins with ^FD
- ^FD used for text on its own and text within barcodes, serialization fields, etc…
- ^FS is required at the end of text.

- It's okay to express the whole label as a graphic, but it will be slower to print, resolution may impact barcode scanning – and field alignment can be impacted

```
^XA
^FO30,30^A0,50,50^FDHELLOWORLD!^FS
^FO30,100^BY3^BCN,100^FD123456^FS
^XZ
```



DEMO!

ZPL Design Tools

Zebra DevCon 2023

- **Text Editors**
  - Ultra Edit – Very Good
  - DOS Editor – Very limited
  - BablePad– Great for encodings
  - Notepad (Risky)
  - ZPL renderers – good, but incomplete. Test with a printer.

- **Label Design –ZebraDesigner Essential and ZebraDesigner for Developers**
  - Point it to a local port (C:\output.txt)
  - Let's you quickly design and see output
  - Design – Print – Examine – Change – Print – Examine for change.

- **Printer Drivers – ZebraDesigner Drivers**
  - Point them to a local port (C:\output.txt)
  - Let's you quickly design and see output
  - Design – Print – Examine – Change – Print – Examine for change!

DEMO!

Unicode Character Printing

- Part of the Printer OS Since 2005
- Multiple languages, encodings and scripts supported
- Printer handles character shaping
- Useful Tools:
  - BablePad
  - BabelMap

```
^XA^CI28^CW1,B:TT0003M_.FNT^LL1200
^FO10,115^A1N30,30^FD.يولد جميع الناس أحرارًا متساوين في الكرامة والحقوق
^FS
^FO10,150^A1N30,30^FD وقد وهبوا عقلاً وضميرًا وعليهم أن يعامل بعضهم بعضًا بروح
.الإخاء
^FS
^XZ
```

Supported Languages & Scripts

Albanian	French	Polish
Arabic	German	Portuguese
Azerbaijani	Greek	Romanian
Bulgarian	Hebrew	Russian
Chinese (Traditional)	Hindi	Serbian
Chinese (Simplified)	Hungarian	Slovak
Croatian	Icelandic	Slovene
Czech	Indonesian	Spanish
Danish	Italian	Swedish
Dutch	Japanese	Tajik
English	Kazakh	Thai
Estonian	Malay	Turkish
Farsi	Moldavian	Ukrainian
Finnish	Korean	Urdu
	Norwegian	Vietnamese

Supported Encodings

Big5	UTF-8
GB2312	UCS-2
Shift JIS	UTF-16
JIS	Wansung
TIS	Johab
GB18030-2000	Unified Hangul Code
Big5HKSCS	Zebra Code Page 850
	Zebra Code Page 1252

DEMO!

~HS – the “Ping” of ZPL

- A Printer Status command, great for feedback
- Considered by Zebra as a backwards compatibility “MUST”.
- Contain a lot of data, but frequently unparsed, used more like “ping”
- Returns three data strings.
- Key data you should consider using:
  - Printer Status
  - Print Buffer status

String 1:

<STX>aaa,**b**,**c**,dddd,**eee**,f,g,h,iii,j,k,l<ETX><CR><LF>

b = paper out flag (1 = paper out)

c = pause flag (1 = pause active)

eee = number of formats in receive buffer

String 2:

<STX>mmm,n,**o**,**p**,**q**,r,s,t,uuuuuuuu,v,www<ETX><CR><LF>

o = head up flag (1 = head in up position)

p = ribbon out flag (1 = ribbon out)

String 3:

<STX>xxxx,y<ETX><CR><LF>

q = thermal transfer mode flag
(1 = Thermal Transfer Mode selected)

DEMO!

RFID Tag Encoding

Zebra DevCon 2023

- Part of the Printer OS since 2000
- Encoding capabilities vary by printer model – use the printer manual!
- Encoding is done using ZPL commands.
- Best Practice is to place the RFID commands at the end of the format.
- Let's encode a Gen 2 RFID Tag in Hexadecimal:

ZPL Commands	Command Functions
^XA	Start of format
^FO50,50 ^A0N,65 ^FDSimple write example ^FS	Prints “Simple write example” on the label at location 50,50.
^RFW,H ^FD112233445566778899001122 ^FS	W,H = write hex Encodes the 12 bytes of data (96 bits) to the tag. The data written is: 112233445566778899001122
^XZ	End of format

RFID Tag Encoding Continued

Zebra DevCon 2023

- Let's Encode the RFID tag, read the tag, and then print the tag data on label:

ZPL Commands	Command Functions
^XA	Start of format
^FO60,60 ^A0N,40 ^FN7 ^FS	When the label prints, the data read from the tag at field variable 7 (^FN7) will be printed at location 60,60.
^RFW,A ^FD0data ^FS	W,A = write ASCII Encodes 0data into the block in hexadecimal format, padded with 8 bytes of zeroes to make the data 12 bytes. The data written is: 306461746100000000000000
^FN7 ^RFR,A ^FS	R,A = read ASCII Reads the tag data into field variable 7 (^FN7). After this occurs, any fields in this label format that have ^FN7 will be replaced with this read data. Because ASCII format was specified, the hexadecimal value is converted back to ASCII format before being printed on the label.
^XZ	End of format

RFID Tag Encoding Continued

Zebra DevCon 2023

- Let's Encode the RFID tag, read the tag, print the data & then return the read result to the host

ZPL Commands	Command Functions
^XA	Start of format
^FO50,50 ^A0N,65 ^FN3 ^FS	When the label prints, the data read from the tag at field variable 3 (^FN3) will be printed at location 50,50.
^RFW,H ^FD0102030405 ^FS	W,H = write hex Encodes 12 bytes of data (96 bits) to the tag with 7 bytes of zeroes as padding. The data written is: 010203040500000000000000
^FN3 ^RFR,H ^FS	R,H = read hexadecimal Reads the tag data into field variable 3 (^FN3). After this occurs, any fields in this label format that have ^FN3 will be replaced with this read data.
^HV3	Returns the value in ^FN3 to the host computer. Data is sent over whichever communication channel is established with the host (such as parallel, serial, USB, Ethernet). In this example, 010203040500000000000000 would be returned to the host
^XZ	End of format

ZPL Best Practices

Zebra DevCon 2023

- Separate ^ and ~ commands. Formatting & encoding commands have priority. Do this instead:

```
~HS
^XA
^FO30,30^A0,50,50^FDHELLOWORLD!^FS
^FO30,100^BY3^BCN,100^FD123456^FS
^XZ
~HS
```

- Separate printer configuration commands from formatting commands. Do this instead:

```
^XA^PR6^XZ
^XA
^FO30,30^A0,50,50^FDHELLOWORLD!^FS
^FO30,100^BY3^BCN,100^FD123456^FS
^XZ
```

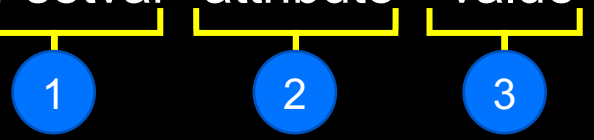
Using Set-Get-Do

Zebra DevCon 2023

- Part of the printer OS since 1995
- A printer configuration and status checking language. Not used for printing

- Three main components
  - Setvar - used to configure printer settings
  - Getvar – used to retrieve printer settings
  - Do – used to command printer to take an action

- Commands must be terminated with space character or CR/LF (0x0D, 0x0A)
- Syntax: !U1 setvar "attribute" "value"



1	**Command**—always preceded with ! U1 and must be specified in lower case. Put a space between ! and U1 and a space between ! U1 and the command.
2	**Attribute**—always in double STRAIGHT quotes and in lower case.
3	**Value** —always in double STRAIGHT quotes. Only applicable for setvar & do.

DEMO!

The allcv

Zebra DevCon 2023

- allcv is a legacy command.
- Returns printer settings in an unstructured format.
- Responses are in a “Branch” order – meaning groups of settings are grouped together.
- A quick, but not machine friendly way to get printer status information.
- Sending ! U1 getvar "allcv" will cause the printer to respond with the allcv list

! U1 getvar "allcv"
bluetooth.
bluetooth.discoverable : on , Choices: on,off
bluetooth.friendly_name : 99J192300067
bluetooth.version : 6.2
bluetooth.date : 01/01/2020
bluetooth.local_name : 99J192300067
bluetooth.address : AC:3F:A4:C8:9C:6B

DEMO!

Using JSON

Zebra DevCon 2023

- Part of the Printer OS since 2013
- JSON (JavaScript Object Notation) is an open standard format that uses human- and machine-readable text for device management.
- JSON is a popular open standard for exchanging data objects and is well suited to this task.
- You can use JSON as an alternative to using the SGD (Set-Get-Do). JSON commands must start with {} .
- The main settings channel for JSON is TCP port 9200. If the printer is using line_print mode, be sure to use port 9200.
- Good JSON viewer at https://codebeautify.org/jsonviewer

{"device.friendly_name":null,
"device.company_name":null,
"device.company_contact":null,
"device.location":null}

DEMO!

Using JSON for Configuration

Zebra DevCon 2023

- To do a setvar in SGD you use the format, as in:
  - ! U1 setvar "sgd.name"
  - ! U1 setvar "ip.port"
  - ! U1 setvar "device.location"
- To set a configuration value using JSON:
  - {}{"sgd.name":"value"} sets the variable value to "value"
  - {}{"ip.port":"1234"} sets the variable value to "1234"
  - {}{"device.location":"my desk"} sets the variable value to "my desk"
- To set several values at once:
  - {}{"device.friendly_name":"XXQLJ120900310",
  - "device.company_contact":"123-555-1212", "device.location":"My Desk"}
- The response is:
  - {"device.friendly_name":"XXQLJ120900310",
  - "device.company_contact":"123-555-1212", "device.location":"My Desk"}

DEMO!

Using JSON for Status

Zebra DevCon 2023

- To do a getvar in SGD you use the format, as in:
  - ! U1 getvar "sgd.name"
  - ! U1 getvar "ip.port"
  - ! U1 getvar "device.location"
- To get a variable value using JSON:
  - {}{"sgd.name":null} returns {"sgd.name":"value"}
    - {}{"ip.port":null} returns {"ip.port":"9100"}
    - {}{"device.location":null} returns {"device.location":"my desk"}
- You can get several values as follows:
  - (){"device.friendly_name":null, "device.company_name":null,
  - "device.company_contact":null, "device.location":null}

  - The response is:
    - {"device.friendly_name":"XXQLJ120900310",
    - "device.company_name":"Zebra Technologies",
    - "device.company_contact":"123-555-1212",
    - "device.location":"My Desk"}

DEMO!

The allconfig

Zebra DevCon 2023

- You can request an allconfig report using JSON, and it will return all printer settings with their current settings, defaults, in JSON format.
- Send this: {}{"allconfig":null}

{"allconfig": {"ip.port":{"**value**":"6101","**type**":"integer","**range**":"0- 65535","**clone**":true,"**archive**":true,"access":"RW"},
"ip.port_alternate":{"**value**":"9100","**type**":"integer","**range**":"0- 65535","**clone**":true,"**archive**":true,"access":"RW"},
"ip.sgd_json_port":{"**value**":"9200","**type**":"integer","**range**":"0- 65535","**clone**":true,"**archive**":true,"access":"RW"}, another setting, ... the last setting}}

“**value**” - the current value stored in the setting
“**type**” - Possible values are integer, enum, bool, string, double, ipv4-address, ipv6- address
“**range**” - The range of a settings. For strings this is the range of the string length. For enums it is the possible enum values.
“**clone**” - Indicates if it is safe to store this setting and apply it to another link-os printer.
“**archive**” - Indicates if is safe to store this setting and apply it to same link-os printer at a later time.
“**access**” - Indicates if the setting is RW (read/write), R (read-only), or W (write-only).

DEMO!

allcv .vs allconfig

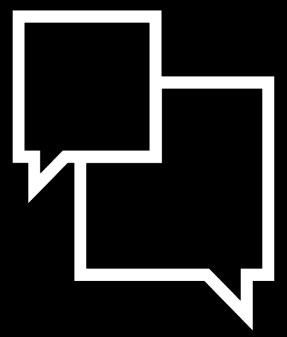
Zebra DevCon 2023

- Which one should you use? Depends on your situation:

  - If you’re creating a new solution, start with Allconfig, it makes dealing with settings and status easer and more consistent.

  - If you’ve embedded Allcv into your solution, keep using it. Consider changing if possible

  - You can use both! Consider coding new features in Allconfig, when changing features that were built using allcv is a challenge



Questions

Zebra DevCon 2023

ZEBRA

Thank You